

VU Research Portal

Low-Variance Policy Gradient Estimation with World Models

Nauman, Michal; Hengst, Floris Den

published in
arXiv.org
2020

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Nauman, M., & Hengst, F. D. (2020). Low-Variance Policy Gradient Estimation with World Models. *arXiv.org*.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:

vuresearchportal.ub@vu.nl

Low-Variance Policy Gradient Estimation with World Models

Michal Nauman
Universiteit van Amsterdam
nauman.mic@gmail.com

Floris den Hengst
Vrije Universiteit Amsterdam
f.den.hengst@vu.nl

Abstract

In this paper, we propose World Model Policy Gradient (WMPG), an approach to reduce the variance of policy gradient estimates using learned world models (WM's). In WMPG, a WM is trained online and used to imagine trajectories. The imagined trajectories are used in two ways. Firstly, to calculate a without-replacement estimator of the policy gradient. Secondly, the return of the imagined trajectories is used as an informed baseline. We compare the proposed approach with AC and MAC on a set of environments of increasing complexity (CartPole, LunarLander and Pong) and find that WMPG has better sample efficiency. Based on these results, we conclude that WMPG can yield increased sample efficiency in cases where a robust latent representation of the environment can be learned.

1 Introduction

Deep reinforcement learning (DRL) has shown impressive results in learning complex behaviours in high-dimensional high-entropy environments ([23]; [32]; [35]). DRL approaches are characterized by the use of neural networks to represent policy, value or other components of the MDP solution. While the end results are impressive, they are often achieved by training that takes days to complete. As such, sample efficiency is an important concern for DRL research.

One of the approaches taken by the community is labeled 'world models' (WMs). In this approach, the agent learns a policy-agnostic representation of the environment, often built by the state, transition and reward networks ([7]; [9]; [14]; [6]). WMs were used to achieve control through planning ([10]; [34]), but also gradient-based policy search, where policy rollouts are imagined by the WM ([14]; and [9]). Besides sample-efficient learning, WMs were shown to facilitate generalization [17] and exploration [29].

Alternatively, better sample efficiency can be achieved by building desirable properties of the estimators used for learning. In the context of the gradient-based policy search, variance of the gradient approximator can be reduced either by advantage [28] or by increasing the amount of samples, often with parallelization ([24]; [1]; [18]). However, the variance reduction as a result of increasing the amount of samples in Monte Carlo (MC) approximation is rapidly diminishing. This is especially true for low entropy policies, where drawing additional independent samples might yield redundant results. To this end, ([27]; [18]) consider approximating the policy gradient by sampling without replacement. The experiments shown promising results, but the requirement of the agent executing multiple trajectories until termination from many states is not scalable to real-world applications.

In this paper, we propose WMPG - a method for approximating low-variance policy gradient. WMPG uses environment interactions to learn a WM in an online fashion. The learned WM allows the policy gradient to be estimated without replacement, with Q-values calculated by finite-horizon forward-looking TD(λ). Two approximation techniques yield a policy gradient with a favourable bias/variance trade-off, while the use of WM makes the method scalable to real-world problems. We

compare the proposed method to AC and MAC on three environments of increasing complexity: CartPole; LunarLander and Pong. We show that the proposed method can achieve better sample efficiency as compared to the benchmark algorithms.

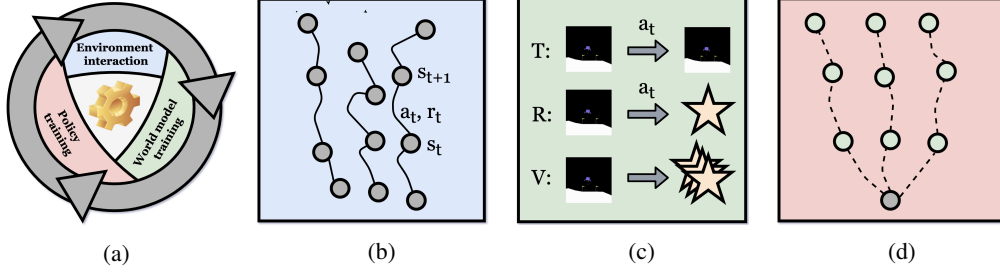


Figure 1: WMPG learning cycle. a) Agent gathers experience to train the WM that is used to estimate the policy gradient. b) Agent samples trajectories from the environment. The learning is triggered once enough experiences are gathered. c) Past transitions are used to train the transition and reward networks. d) For every state in the experience batch, agent samples multiple actions without replacement and imagines the Q-values of those actions.

The main contribution of this paper is unifying no-replacement policy gradient approximation with world model framework. To the best of our knowledge, we are the first to propose: 1) using world models to calculate the policy gradient with a without-replacement expectation estimator; 2) using variable number of without-replacement sampled actions per state. We make our code available at: <https://github.com/WMPG-paper/WMPG>.

The paper is organized as follows. We firstly provide background for the problem that this paper addresses (policy gradient approximation) and the method that this paper uses (WMs). Next, we introduce the proposed approach, by listing the components of WMPG and explaining how WMPG calculates the policy gradient approximator. Finally, we discuss related work and the experiments performed for this paper.

2 Background

Reinforcement Learning RL considers problems in the framework of Markov decision processes (MDPs). We consider deterministic MDPs with finite action sets. Thus, MDP is a tuple $\mathcal{M} : \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ where \mathcal{S} is a continuous state representation, \mathcal{A} is a finite set of actions with size $|\mathcal{A}|$, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is a deterministic transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function and $\gamma \in [0, 1]$ is a discount factor to balance current and future rewards. After observing state s_t at each time step t , the agent performs action a_t up to some end time T , following policy $\pi \in \Pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ and collects reward $r_t = \mathcal{R}(s_t, a_t)$ at every time-step. The goal is to find a policy that maximizes the sum of collected rewards over time.

Policy Gradient Methods Gradient-based policy search defines a policy function π_θ that is differentiable wrt. to its parameters θ [36]. Then, θ is optimized such that the expected value of starting states is maximized, with expectation taken wrt. to the starting states. Assuming a single starting state, the gradient of the objective can be expressed as ([33]; [24]):

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \pi_\theta} [\nabla_\theta J(\theta, s)] = \mathbb{E}_{s \sim \pi_\theta} \left[\sum_{a \in \mathcal{A}} Q^{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a|s) \right] \quad (1)$$

By applying the log-derivative trick, the inner sum can be rewritten as expectation, such that policy gradient at state s^* becomes equal to:

$$\nabla_\theta J(\theta, s^*) = \mathbb{E}_{a \sim \pi_\theta} [Q^{\pi_\theta}(s^*, a) \nabla_\theta \log \pi_\theta(a|s^*)] \quad (2)$$

In most practical applications, agent can perform only one action before being transitioned to some further state. As a result, the above expectation is often evaluated with single-sample MC, such

that $\nabla_{\theta} \hat{J}(\theta, s^*) = Q^{\pi_{\theta}}(s^*, a) \nabla_{\theta} \log \pi_{\theta}(a|s^*)$ with $a \sim \pi_{\theta}$ ([28]; [24]; [18]). Given single-sample estimation with some action a_i , the sign of $\hat{J}(\theta, s^*)$ depends solely on the sign of $Q^{\pi_{\theta}}(s^*, a_i)$.

Furthermore, $Q^{\pi_{\theta}}(s^*, a)$ might itself be unknown. If so, $\hat{Q}^{\pi_{\theta}}(s^*, a)$ can be calculated simultaneously with $\nabla_{\theta} \hat{J}(\theta, s^*)$ using value approximation techniques like MC policy rollout, TD(n) or TD(λ) ([28]; [5]; [9]). Given some method of Q-value approximation, the non-zero variance of $\nabla_{\theta} \hat{J}(\theta, s^*)$ can be reduced with an additive control variate $b(s)$. Then, the term $(\hat{Q}^{\pi_{\theta}}(s^*, a) - b(s))$ is referred to as advantage. Having calculated $\nabla_{\theta} \hat{J}(\theta, s)$ for every state in batch D_s , the scalar loss is calculated with the batch average $\nabla_{\theta} \hat{J}(\theta) = \frac{1}{|D_s|} \sum_{s \in D_s} \nabla_{\theta} \hat{J}(\theta, s)$.

World Models The framework of world models is designed to achieve optimal control with learned components of the underlying MDP. WMs are often represented by state embedding, transition and reward networks ([7]; [14]; [9]; [34]). Since successful control largely depends on the problem representation, a lot of attention is focused on learning latent representations that facilitate policy search ([6]; [17]). Particularly, two classes of methods seem to emerge from the recent literature: reconstruction ([7]; [13]; [10]; [9]; [14]); and bisimulation approximation ([17]; [6]; [34]).

In reconstruction, original state representation is auto-encoded and the representation loss is placed in the original problem space. Thus, the state embedding is trained independently of the policy loss and other WM components. This method is known to have failure modes, especially when defined in the pixel space. Latent representations learned through reconstruction were shown to ignore important objects that are visually small (ie. ball in Pong) or use model capacity on rich backgrounds, which might be irrelevant from the perspective of state value ([14]; [17]).

Bisimulation is guaranteed to be value preserving ([20]; [21]). Bisimulation can be approximated with bisimulation loss [6]. As the objective is placed outside of the pixel space, the approach directly tackles reconstruction’s modes of failure. However, it was shown to map all states to a single point for sparse reward environments [17]. The proposed solution, contrastive bisimulation [34], was not tested in the context of on-policy WM learning.

Learning robust latent state representations is orthogonal to this paper. Recent research has shown approaches of learning latent representation for RL ([7]; [10]; [10]; [14]; [38]; [2]; [4]). Contrary to that work, this paper shows how learned latent models can be used to approximate low-variance policy gradient.

3 World Model Policy Gradient

WMPG learns a world model and uses it to approximate $\nabla_{\theta} J(\theta, s)$ for every state in the data batch. To calculate $\nabla_{\theta} J(\theta, s)$, WMPG samples multiple actions without replacement and imagines trajectories of length h starting with those actions. Q-values of those actions are calculated using TD(λ) calculated over the imagined trajectories. Finally, the without-replacement value estimator is used as a baseline. WMPG uses two components with independent memory buffers: a WM with a state embedding, a transition and a reward networks; and a behavior model consisting of a policy and a value networks.

State embedding $Z_{\mu} : \mathcal{S} \rightarrow \mathcal{Z} \in \mathbb{R}^m$ maps the original state representation into some smaller dimensionality of size m . State dimensionality reduction allows the agent to learn more robust reward and transition approximators, as well as reduce the complexity of the policy search. Learning optimal state representation in the context of RL is an open research topic. As such, we learn the state compression with variational reconstruction in the pixel space, which was found to be working relatively well in multiple tasks.

Transition network $T_{\kappa} : \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{Z}' \in \mathbb{R}^m$ approximates the transitions of the underlying MDP. Given that the underlying MDP is deterministic, transition can be modelled with a feedforward neural network. Transition function parameters κ are trained with mean absolute distance loss against the compressed transitions from the environment.

Reward network $R_o : \mathcal{Z} \times \mathcal{A} \rightarrow \mathbb{R}$ maps state-action pairs to rewards. Similarly to the transition function, R_o is trained with mean absolute distance against the real rewards from the environment.

Policy network $\pi_\theta : \mathcal{Z} \rightarrow \mathcal{P}(\mathcal{A}|\mathcal{Z})$ outputs a discrete probability distribution over all actions in a given state, given the compressed latent representation of that state.

Value network $V_\phi : \mathcal{Z} \rightarrow \mathbb{R}$ learns state values under policy π_θ . Similarly to actor-critic algorithms, the value network can be supervised with a variety of techniques. For simplicity, we choose MC policy rollout values (ie. rewards gathered during the previous episode).

Memory modules Behaviour memory stores only the most recent trajectories and is used as a basis for learning the policy and value approximators. Similarly to an experience buffer [22], WM memory stores some fixed amount of recent transitions, which do not have to be distributed according to the policy. Those policy-agnostic transitions are used to train the WM components of the agent.

Algorithm 1 WMPG episode

```

1: function WMPG EPISODE()
2:    $s = env.reset()$     % reset the environment
3:   while not terminal do
4:      $z = Z_\mu(s)$     % compress the state representation using  $Z_\mu$ 
5:      $a \sim \pi_\theta(z)$     % sample an action from the current policy
6:      $s', r = env.step(a)$  % execute the sampled action in the environment
7:      $z' = Z_\mu(s')$     % compress the future states before adding experience to the memory
8:      $\mathcal{D}_\Pi \leftarrow (z, a, r, z')$  % add experience to the on-policy memory
9:     if  $len(\mathcal{D}_\Pi) > batch\ size$  then
10:       $\mathcal{D}_{WM} \leftarrow \mathcal{D}_\Pi$  % add the on-policy data to the WM memory
11:      while learning iterations do
12:         $\dot{z}, \dot{a}, \dot{r}, \dot{z}' \sim \mathcal{D}_{WM}$  % sample transitions for WM training
13:         $\kappa \leftarrow \kappa - \alpha \nabla_\kappa |T_\kappa(\dot{z}, \dot{a}) - \dot{z}'|$  % train the transition network
14:         $o \leftarrow o - \alpha \nabla_o |R_o(\dot{z}, \dot{a}) - \dot{r}|$  % train reward the network
15:         $\phi \leftarrow \phi - \alpha \nabla_\phi |V_\phi(z) - V(\mathcal{D}_\Pi)|$  % train the value network
16:         $\theta \leftarrow \theta + \alpha \nabla_\theta \hat{J}(\theta, z)$  % train the policy using steps described in Section 4
17:      end while
18:       $\mathcal{D}_\Pi = []$  % wipe the on-policy memory
19:    end if
20:  end while
21: end function

```

Figure 2: WMPG learning episode. Unrolling the policy in imagination allows for multiple parameter updates using one batch of the on-policy data \mathcal{D}_Π . For detailed implementation, we point the reader to: <https://github.com/WMPG-paper/WMPG>.

4 Gradient Estimation in WMPG

WMPG approximates $\nabla_\theta J(\theta, s)$ using k actions sampled without replacement from the world model. For any $k \leq |\mathcal{A}|$, $\nabla_\theta \hat{J}(\theta, s)$ can be calculated by without-replacement MC estimator ([12]; [27]; [30]; [18]):

$$\nabla_\theta \hat{J}(\theta, s) = \sum_{i=1}^k \frac{\pi_\theta(a_i|s)}{\Omega(a_i|\pi_\theta, k, s)} Q^{\pi_\theta}(s, a_i) \nabla_\theta \log \pi_\theta(a_i|s) \quad (3)$$

Where i indicates the concurrent actions sampled without replacement from the WM and $\Omega(a_i|\pi_\theta, k, s)$ is the inclusion probability of action a_i (ie. probability that action a_i was sampled without replacement given k samples and policy π_θ in state s). For any fixed k the estimator is an unbiased approximator of the exact expectation, which we show in Appendix A. Conveniently, for $k = 1$ the estimator becomes equal to a vanilla MC estimator, while for $k = |\mathcal{A}|$ the estimator

becomes equal to the expected value over the sampled domain. Figure 3 shows how the WM is used to calculate the without-replacement approximator.

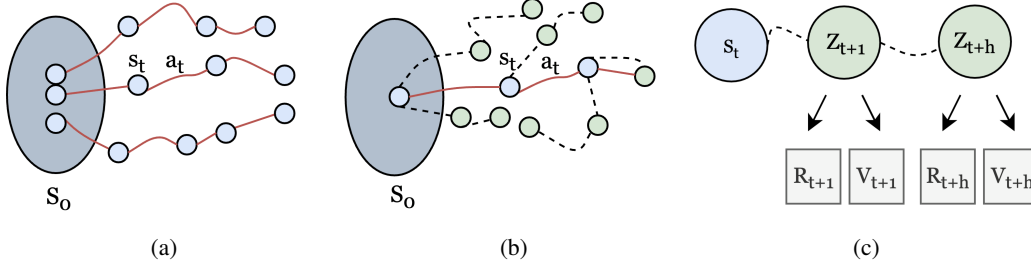


Figure 3: WMPG combines acting in the environment with learning in the imagination. a) Trajectories are sampled from the environment (represented by red lines) and gathered into a batch. b) Agent decides how many without-replacement actions should be sampled per batched state and imagines a fixed-length trajectories for each sampled action. c) Q-values of the sampled actions are approximated using $TD(\lambda)$.

Intuitively, the without-replacement estimator allows for efficient evaluation of the expectation by not oversampling the high-probability domain. If the policy has low entropy, then sampling with replacement can yield repeated results, which in the context of policy gradient is uninformative.

Formally, the estimator reduces the approximation variance by remodelling the policy gradient from the sum of independent random variables to the weighted sum of dependent random variables. This slows the diminishing of the variance reduction associated with drawing additional samples. For details, we refer the reader to Appendix B.

In the following paragraphs, we detail how the policy gradient is approximated in WMPG. We describe the process in four steps: 1) *choosing a k* ; 2) *Q-value approximation*; 3) *baseline calculation*; and 4) *weight normalization*.

Choosing k The without-replacement estimator is unbiased for any fixed size of k . Since $\nabla_{\theta} J(\theta, s)$ is approximated for every batched state independently, k can be chosen per evaluated state. Thus, it seems natural to ask: is there a way of choosing a k that well-balances the robustness of the estimator with the required compute? While sampling theory offers frameworks to tackle such questions ([11]; [30]), we consider the following simple heuristics: constant k throughout the training; k decreasing throughout the training; and k dependent on the entropy of policy at given state.

Q-value approximation WMPG uses the transition, reward and value networks to approximate the unknown Q-values. For each sampled action, WMPG imagines a trajectory of length h and estimates the respective Q-value with $TD(\lambda)$:

$$\hat{Q}^{\pi_{\theta}}(s, a, \lambda, h) = (1 - \lambda) \sum_{n=1}^{h-1} \lambda^{n-1} TD(n|s, a) + \lambda^{h-1} TD(h|s, a) \quad (4)$$

Where $TD(n|s, a)$ denotes n -step temporal difference value, given the starting state-action pair and following policy π_{θ} .

Baseline Variance Reduction WMPG uses the without-replacement value approximation as a baseline for the policy gradient:

$$\hat{V}(s) = \sum_{i=1}^k \frac{\pi_{\theta}(a_i|s)}{\Omega(a_i|\pi_{\theta}, k, s)} \hat{Q}^{\pi_{\theta}}(s, a_i, \lambda, h) \quad (5)$$

Since the importance weights depend on the sampled actions, the resulting estimator becomes biased. Such bias can be corrected by introducing an action-dependent term $C(s, a_i) = (1 + \frac{\pi_{\theta}(a_i|s)}{\Omega(a_i|\pi_{\theta}, k, s)}) -$

$\pi_\theta(a_i|s)$) [18]:

$$\nabla_\theta \hat{J}(\theta, s) = \sum_{i=1}^k \frac{\pi_\theta(a_i|s)}{\Omega(a_i|\pi_\theta, k, s)} \left(C(s, a_i) \hat{Q}^{\pi_\theta}(s, a_i, \lambda, h) - \hat{V}(s) \right) \nabla_\theta \log \pi_\theta(a_i|s) \quad (6)$$

When all actions are sampled, then the baseline becomes independent of the sampling process. As such, if $k = |\mathcal{A}|$ then $\frac{\pi_\theta(a_i|s)}{\Omega(a_i|\pi_\theta, k, s)} = \pi_\theta(a_i|s)$, and in consequence $C(s, a_i) = 1$ for all i .

Weight normalization The weighted probabilities do not generally sum to one. While such calculation leads to unbiased estimation, it has high variance. As the policy gradient is typically evaluated once before the parameter update, we are interested in low-variance approximation of $\nabla_\theta \hat{J}(\theta, s)$. The without-replacement expectation estimator can be normalized by the sum of the sampled weights [19]. To this end, we incorporate the normalization into the state policy gradient calculations according to:

$$\nabla_\theta \hat{J}(\theta, s) = \sum_{i=1}^k \frac{\pi_\theta(a_i|s)}{\Omega(a_i|\pi_\theta, k, s)} \left(\frac{\hat{Q}^{\pi_\theta}(s, a, \lambda, h)}{W_i(s)} - \frac{\hat{V}(s)}{W(s)} \right) \nabla_\theta \log \pi_\theta(a_i|s) \quad (7)$$

With $W(s) = \sum_{i=1}^k \frac{\pi_\theta(a_i|s)}{\Omega(a_i|\pi_\theta, k, s)}$ and $W_i(s) = W(s) - \frac{\pi_\theta(a_i|s)}{\Omega(a_i|\pi_\theta, k, s)} + \pi_\theta(a_i|s)$. While the normalization introduces bias, the estimator remains consistent [11] and is reported to perform better in the context of REINFORCE [19].

5 Experiments

We conducted experiments to test sample efficiency (ie. number of the required environment episodes to achieve a certain performance) of the proposed approach as compared to standard policy gradient algorithms. We test the WMPG sample efficiency in two contexts: 1. *compact state representation*; and 2. *high-dimensional state representation*. To limit the scope of the hyperparameter search, we do not search for WMPG hyperparameters that are available for AC. For those parameters, WMPG uses the values found for the AC implementation. The results are presented on Figure 4; for further details regarding the experimental setting, see Appendix C.

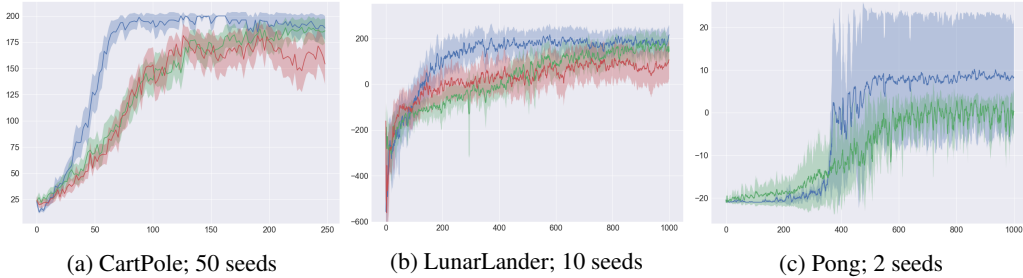


Figure 4: Learning curves. y -axis denotes the average accumulated reward and x -axis denotes the environment episode number. [Blue] is WMPG; [green] is AC; and [red] is MAC. Shadow denotes the distance of two standard deviations of the average. a) and b) Given compact value preserving state representation, WMPG yields robust performance gain. c) Encoding the state representation with a VAE can result in faster policy convergence as compared to learning on the original problem representation ([7]; [10]). In our experiments, only one seed converged to the optimal policy of 21 points per episode. Thus, we hypothesize that learning control via VAE compressed representations might be more prone to local optima with suboptimal policies.

Compact state representation We treat the problem of representation learning in WM as orthogonal to this paper. As such, we test WMPG in environments where representation learning can be omitted (ie. state embedding function can be represented by an identity mapping) and the state representation is guaranteed to be value preserving. By doing so, we look for the sample efficiency

gains which are directly attributable to mechanisms of WMPG, that is without-replacement approximation of $\nabla_{\theta} J(\theta, s)$ and $\text{TD}(\lambda)$ approximation of Q-values using online-learned transition and reward mappings. For the compact state representation testing we choose the CartPole and LunarLander environments, run for respectively 250 and 1000 environment episodes.

High-dimensional state representation Further, we test WMPG against AC on high-dimensional visual input of Arcade Learning Environment (ALE) Pong [3] given 1000 environment episodes. Here, we learn the representations with a variational reconstruction loss placed in the pixel space [16]. The VAE is pre-trained using data generated over 10 episodes following a random policy. Learning on a compressed latent representation effectively decreases the size of policy search problem and is expected to yield efficiency gains independently of other mechanisms used. Therefore, to limit the impact that state embedding has on the results, we train AC on the VAE state representations used for WMPG.

Ablations WMPG integrates several mechanisms that could potentially lead to better sample efficiency. To gain a better understanding of the contribution of each mechanism towards the final performance, we performed additional experiments. Firstly, we look how sampling additional actions without replacement effects the WMPG performance (Figure 5). Furthermore, we investigate the learning curves of WMPG given $k = |\mathcal{A}|$ and different values of h and λ (Figure 6).

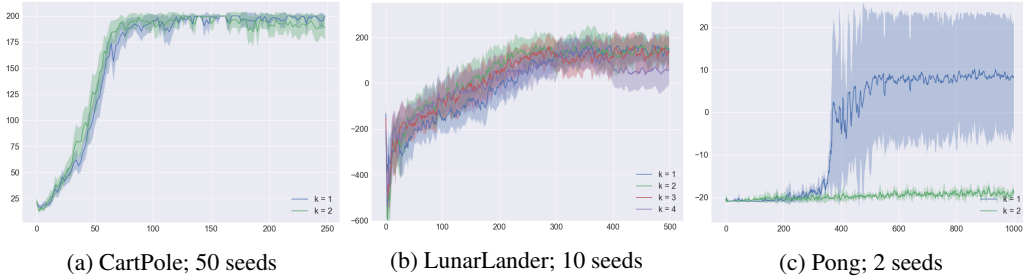


Figure 5: WMPG performance for different number of without-replacement actions. y -axis denotes the average accumulated reward and x -axis denotes the environment episode number. [Blue] is $k = 1$; [green] is $k = 2$; [red] is $k = 3$; and [purple] is $k = 4$. If only one action is sampled, then we use the value network at given state as the baseline. Shadow denotes the distance of two standard deviations of the average. a) and b) Q-values of the imagined actions are biased due to $\text{TD}(\lambda)$ and WM components. Therefore, imagining additional actions with a WM might induce a bias-variance trade-off, especially in the early stages of training. c) Imagining all actions yielded almost random results on all tested Pong seeds. We explain this with the fact that WMPG nullifies the gradient when Q-values of different action are equal (look Equation 6). Since Pong has relatively sparse rewards, it might be that short-horizon $\text{TD}(\lambda)$ approximations yield similar Q-values for many actions at a given state.

6 Related Work

World models VPN simulates multi-step returns to approximate and backup the Q-values over a horizon [25]. Ha et al. [7] uses evolutionary approaches to search for policy in the compact WM representation of the environment. Similarly, PLANET [10] uses non-differentiable optimization to plan within the learned WM. Value iteration was successfully applied to learned WM representations [34]. MuZero [26] leverages the learned WM to perform Monte-Carlo tree search in the latent representation. SimPLe [14] and Dreamer [9] calculate the policy gradient with trajectories imagined by the WM.

Policy gradients A3C leverages parallel agents to gather many trajectories from the environment, but calculates the policy gradient with MC [24]. TRPO vine procedure [27] and [19] consider sampling actions without replacement at sampled states. There, agent is allowed to ‘rewind’ back to a given state. Kool et al. propose to use the without-replacement samples to calculate an informed baseline [18]. MAC uses Q-network approximations to calculate $\nabla_{\theta} J(\theta, s)$ exactly (ie. iterating

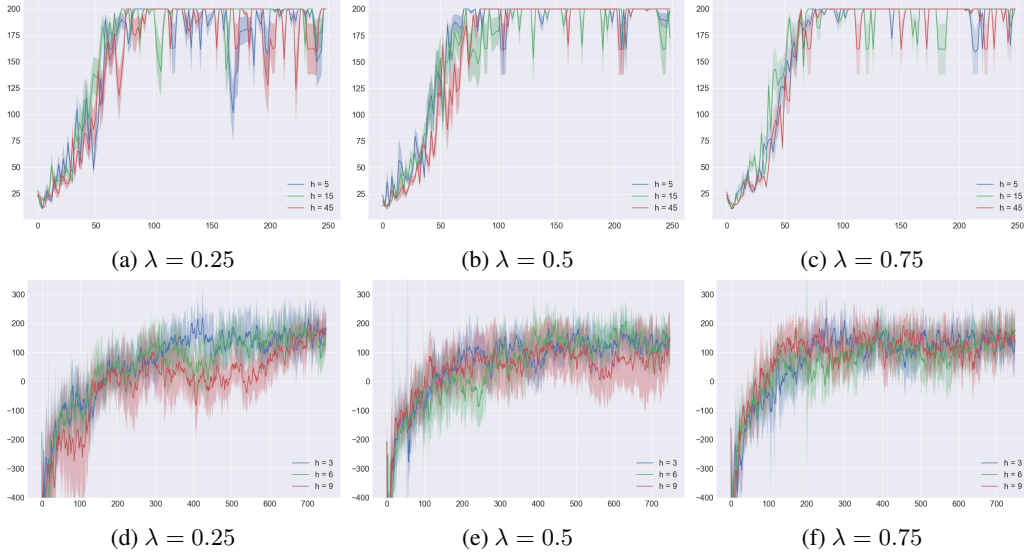


Figure 6: WMPG learning curves given different horizon lengths and λ . y -axis denotes the average reward accumulated over 8 seeds and x -axis denotes the episode number. Colours denote different imagination horizon lengths. Shadow denotes the distance of two standard deviations of the average. a), b), c) depict CartPole. There, [blue]: $h = 5$; [green]: $h = 15$; and [red]: $h = 45$. d), e) and f) depict LunarLander. There, [blue]: $h = 3$; [green]: $h = 6$; and [red]: $h = 9$.

over all actions) [1]. TRPO [27] updates policy parameters with natural gradient [15], but was found to be costly for bigger models. ACKTR [37] calculates the trust-region update using less costly Kronecker-factored approximation.

7 Conclusion

Imagining trajectories with latent models allows for gradient approximation strategies unavailable to traditional agents. In particular, we showed that WM can be used to sample and evaluate many actions per analyzed state. Contrary to MC, the without-replacement expectation estimator does not redundantly resample the-high probability domain. Our results indicate that WM-based without-replacement trajectory sampling is a promising approach for better sample efficiency in gradient-based policy search.

Ablation studies have revealed a surprisingly nonlinear relation between the number of actions sampled without replacement and the agent performance in the context of WM-based learning. Imagining trajectories indeed reduces variance of policy gradient, but at the cost of bias that is inherent to WM approximations. Thus, we suspect that increasing the number of sampled action creates a bias-variance trade-off, with bias magnified by each imagined trajectory. Furthermore, our results indicate that varying the number of imagined samples yields better performance than any constant number of samples. We believe that applying recent advances in incremental without-replacement sampling ([30]; [31]) to gradient-based policy search might be a promising research direction.

WMPG is an approach for sample efficient reinforcement learning using world models. The approach learns a world model which is used to imagine trajectories sampled without replacement. These are used to estimate policy gradients with a low variance without-replacement expectation estimator. We showed that the proposed approach can yield increased sample efficiency as compared to AC and MAC.

Broader Impact

This paper considers the problem of variance reduction in policy gradient approximation. As such, authors believe that there are no societal consequences specific to this work.

References

- [1] Asadi, K., Allen, C., Roderick, M., Mohamed, A.-r., Konidaris, G., Littman, M., and Amazon, B. U. Mean actor critic. *stat*, 1050:1, 2017.
- [2] Baradel, F., Neverova, N., Mille, J., Mori, G., and Wolf, C. Cophy: Counterfactual learning of physical dynamics. In *International Conference on Learning Representations*, 2020.
- [3] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.
- [4] Biza, O., Platt, R., van de Meent, J.-W., and Wong, L. L. Learning discrete state abstractions with deep variational inference. *arXiv preprint arXiv:2003.04300*, 2020.
- [5] Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.
- [6] Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *ICML*, 2019.
- [7] Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pp. 2450–2462, 2018.
- [8] Haarnoja, T. *Acquiring Diverse Robot Skills via Maximum Entropy Deep Reinforcement Learning*. PhD thesis, UC Berkeley, 2018.
- [9] Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.
- [10] Hafner, D., Lillicrap, T. P., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *ICML*, 2019.
- [11] Hesterberg, T. C. *Advances in importance sampling*. PhD thesis, Citeseer, 1988.
- [12] Horvitz, D. G. and Thompson, D. J. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.
- [13] Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pp. 2117–2126, 2018.
- [14] Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osinśki, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2019.
- [15] Kakade, S. M. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.
- [16] Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.
- [17] Kipf, T., van der Pol, E., and Welling, M. Contrastive learning of structured world models. In *International Conference on Learning Representations*, 2019.
- [18] Kool, W., van Hoof, H., and Welling, M. Buy 4 reinforce samples, get a baseline for free! 2019.
- [19] Kool, W., Van Hoof, H., and Welling, M. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pp. 3499–3508, 2019.
- [20] Larsen, K. G. and Skou, A. Bisimulation through probabilistic testing. *Information and computation*, 94(1):1–28, 1991.
- [21] Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

- [22] Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
- [23] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [24] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.
- [25] Oh, J., Singh, S., and Lee, H. Value prediction network. In *Advances in Neural Information Processing Systems*, pp. 6118–6128, 2017.
- [26] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.
- [27] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.
- [28] Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2016.
- [29] Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. *arXiv preprint arXiv:2005.05960*, 2020.
- [30] Shah, R. and Kroese, D. P. Without-replacement sampling for particle methods on finite state spaces. *Statistics and Computing*, 28(3):633–652, 2018.
- [31] Shi, K., Bieber, D., and Sutton, C. Incremental sampling without replacement for sequence models. *arXiv preprint arXiv:2002.09067*, 2020.
- [32] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- [33] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.
- [34] van der Pol, E., Kipf, T., Oliehoek, F. A., and Welling, M. Plannable approximations to mdp homomorphisms: Equivariance under actions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1431–1439, 2020.
- [35] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- [36] Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- [37] Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pp. 5279–5288, 2017.
- [38] Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M. J., and Levine, S. Solar: Deep structured representations for model-based reinforcement learning. In *ICML*, 2019.

A Unbiasedness of the without-replacement expectation estimator

We assume X to be a discrete random variable with domain of $|\mathcal{A}|$ values $(x_1, \dots, x_{|\mathcal{A}|})$ and respective probabilities $(p_1, \dots, p_{|\mathcal{A}|})$. The expected value of X is given by:

$$\mathbb{E}[X] = \sum_{a=1}^{|\mathcal{A}|} p_a x_a \quad (8)$$

We denote \hat{x} as the following estimator of $\mathbb{E}[X]$:

$$\hat{x} = \sum_{i=1}^k \beta_i x_i \quad (9)$$

Where k is the number of without-replacement samples drawn from X , x_i is the i^{th} sample drawn without replacement and β_i is a weight independent of determined by x_i . We want to show that for any constant value of $k \leq |\mathcal{A}|$, \hat{X} is unbiased is expectation.

$$\mathbb{E}[\hat{X}] = \mathbb{E}[X] \quad (10)$$

We first note that \hat{X} is defined as k without-replacement samples from X . Therefore, \hat{X} has a domain of $\binom{\mathcal{A}}{k} = K$ values denoted as $\hat{X} = (\hat{x}_1, \dots, \hat{x}_K)$ and respective probabilities (q_1, \dots, q_K) . Therefore:

$$\mathbb{E}[\hat{X}] = \sum_{j=1}^K q_j \hat{x}_j = \sum_{j=1}^K q_j \sum_{i=1}^k \beta_i x_i = \sum_{j=1}^K q_j \beta_1 x_1^j + q_j \beta_2 x_2^j + \dots + q_j \beta_k x_k^j \quad (11)$$

Now, we note that x_a can be sampled in some number of \hat{x} . We denote the set of indices of \hat{X} that contain x_a as L . Then, by expanding the sum:

$$\mathbb{E}[\hat{X}] = \sum_{a=1}^{|\mathcal{A}|} \beta_a x_a \sum_{j \in L} q_j \quad (12)$$

The above operation explicitly assumes that β_a is independent of j . Combining Equations 8, 10 and 12 yields:

$$\sum_{a=1}^{|\mathcal{A}|} p_a x_a = \sum_{a=1}^{|\mathcal{A}|} \beta_a x_a \sum_{j \in L} q_j \quad (13)$$

Thus, when $\beta_a = \frac{p_a}{\sum_{j \in L} q_j}$, the estimator is an unbiased estimator of $\mathbb{E}[X]$.

B Variance of the without-replacement policy gradient estimator

Here, we show why calculating policy gradient without replacement might lead to a better variance reduction stemming from increase of samples. To not overload the notation, we assume the Q-values to be known.

Given a batch of states $D_s = (s_1, s_2, \dots, s_{|D_s|})$, gradient $\nabla_{\theta} \hat{J}(\theta)$ is calculated with batch average. Without assuming independence between sampled states, variance of the policy gradient estimator is equal to:

$$Var[\nabla_{\theta} \hat{J}(\theta)] = \frac{1}{|D_s|^2} \sum_{s \in D_s} \left(Var[\nabla_{\theta} \hat{J}(\theta, s)] + \sum_{s' \neq s \in D_s} Cov[\nabla_{\theta} \hat{J}(\theta, s), \nabla_{\theta} \hat{J}(\theta, s')] \right) \quad (14)$$

Where $\nabla_{\theta} \hat{J}(\theta, s)$ denotes evaluation of the policy gradient at state s . If $\nabla_{\theta} \hat{J}(\theta, s)$ is calculated via MC with $|D_a|$ samples, then:

$$Var[\nabla_{\theta} \hat{J}^{mc}(\theta, s)] = \frac{Var[Q^{\pi_{\theta}}(s, a) \nabla_{\theta} \log \pi_{\theta}(a|s)]}{|D_a|} \quad (15)$$

Taking derivative wrt. $|D_a|$ reveals that the decrease of variance stemming from taking more samples is rapidly diminishing. Furthermore, if the variance is non-zero, then $Var \left[\nabla_{\theta} \hat{J}^{mc}(\theta, s) \right]$ converges to 0 for $|D_a| = \infty$.

If we treat $|D_a|$ as the number of actions sampled without replacement, then for $|D_a| = |\mathcal{A}|$ (ie. when sampling all actions) $\nabla_{\theta} \hat{J}(\theta, s)$ is calculated with exact expectations:

$$\nabla_{\theta} \hat{J}^{wr}(\theta, s) = \sum_{a \in |\mathcal{A}|} \nabla_{\theta} \pi_{\theta}(a|s) Q^{\pi_{\theta}}(s, a) \quad (16)$$

Thus, given $|D_a| = |\mathcal{A}|$ and known Q-values, it follows that $Var \left[\nabla_{\theta} \hat{J}^{wr}(\theta, s) \right] = 0$. As such, assuming $|D_a| < |\mathcal{A}|$ it might be the case that $Var \left[\nabla_{\theta} \hat{J}^{wr}(\theta, s) \right] < Var \left[\nabla_{\theta} \hat{J}^{mc}(\theta, s) \right]$. Borrowing the notation from Appendix A, the variance of without-replacement estimator is equal to:

$$Var \left[\nabla_{\theta} \hat{J}^{wr}(\theta, s) \right] = \sum_{j=1}^K q_j \left(\sum_{i=1}^{|D_a|} \beta_i x_i \right)^2 - (\nabla_{\theta} J(\theta, s))^2 \quad (17)$$

For further reading on the without-replacement estimation variance, we point the reader to ([12]; [19]; [30]).

C Experiment details

Below, we detail the implementation of the experiments. Table 1 lists the searched hyperparameters for each model:

Name	Description	AC	MAC	WMPG
π_{size}	Policy network size	x	x	x
V_{size}	Value network size	x	(x)	x
T_{size}	Transition network size			(x)
R_{size}	Reward network size			(x)
O_{π}	Optimizer for π	(x)	(x)	x
O_V	Optimizer for value	(x)	(x)	x
O_T	Optimizer for transition			(x)
O_R	Optimizer for reward			(x)
α_{π}	Learning rate for π	(x)	(x)	x
α_V	Learning rate for value	(x)	(x)	x
α_T	Learning rate for transition			(x)
α_R	Learning rate for reward			(x)
I_G	General learning iterations		(x)	(x)
I_V	Value learning iterations		(x)	(x)
I_G	WM learning iterations			(x)
h	Horizon length			(x)
k	Sampled actions			(x)
λ	TD(λ) controller			(x)
CartPole		81	126	126
# of configs	LunarLander	9	24	36
	Pong	8	NA	4

Table 1: Number of tunable hyperparameters; x denotes that parameter is tunable; (x) denotes that parameter was tuned for experiments in this paper

Both AC and MAC are implemented using two separate feedforward neural networks. Both value and Q-value networks are trained using MC policy rollout.

All models use a discount factor of 0.99 and batch sizes of (CartPole - 32; LunarLander - 64; and Pong - 512). All models run on LunarLander and Pong use a multiplicative learning rate annealing with a rate of 0.99 and a step sizes of 125 and 5 respectively.

Pong environment is implemented with a frame-skip of four and non-sticky actions. Finally, the Pong frames are preprocessed: size of the frame is decreased to (80, 80, 1) and the ball is enlarged by the factor of two.

For further details regarding the implementation, we point the reader to the provided GitHub repository. Finally, all Pong agents use an entropy coefficient of 0.01 [8].

C.1 AC

Table below lists the hyperparameter values used for AC:

env	π_{size}	V_{size}	α_{π}	α_V	O_{π}	O_V	I_G
CartPole	32	64	.0025	.005	RMS	RMS	1
LunarLander	64	64	.0025	.0025	RMS	RMS	1
Pong	512	512	.001	.001	RMS	RMS	1

Table 2: Best performing AC configurations.

C.2 MAC

Table below lists the hyperparameter values used for MAC:

env	π_{size}	Q_{size}	α_{π}	α_Q	O_{π}	O_Q	I_G	I_V
CartPole	32	[64,64]	.00125	.005	RMS	RMS	3	3
LunarLander	64	[64,64]	.0025	.005	RMS	RMS	5	3

Table 3: Best performing MAC configurations.

C.3 WMPG

Table below lists the hyperparameter values used for WMPG:

env	T_{size}/R_{size}	α_T/α_R	k	O_R/O_R	I_G	I_V	I_{WM}	h	λ
CartPole	64	.005	2	Adam	5	3	5	15	0.75
LunarLander	96	.005	Decrease	Adam	4	3	5	3	0.75
Pong	1028	.002	1	Adam	5	1	1	3	0.75

Table 4: Best performing configurations of WMPG. Other settings are taken from respective AC implementations.